1. feladat: ismerkedés a jelölésekkel, katalógusinformációk, algoritmusok használata

Egy bank nyilvántartásában található, SZÁMLA(TELEPÜLÉS, EGYENLEG, ...) sémára illeszkedő relációból szeretnénk megtudni a budapesti számlák adatait. Ezeket tudjuk a relációról:

- $f_{\text{SZÁMLA}} = 40$: SZÁMLA reláció 40 rekordja fér bele egy lemezblokkba.
- $V(\text{TELEPÜLÉS}, \text{SZÁMLA}) = 50$: 50 különböző fiók-név létezik a SZÁMLA relációban.
- $V(\text{EGYENLEG}, \text{SZÁMLA}) = 500$: 500 különböző értékű számla van a SZÁMLA relációban.
- $n_{\text{SZÁMLA}} = 10\ 000$: A SZÁMLA relációnak 10 000 eleme van.

A feladat megoldása során tegyük fel, hogy a reláció elemeit optimális blokk-kihasználtság mellett tároljuk. Kérdések:

(a) Adjuk meg a feladatot megoldó relációs algebrai lekérdezést!

(b) Mennyi minimális/maximális/átlagos költség, ha lineáris keresést alkalmazunk? Mitől függ, hogy mennyi?

(c) Tfh. a rekordok a fiók szerint rendezetten tárolódnak. Mennyi a bináris keresés várható költsége?

2. feladat: a join nagyságának becslése

Adott két relációs sémánk, a BANKBETÉT és az ÜGYFÉL. Illesszük a két (sémára illeszkedő) relációt a mindkettőben szereplő ÜGYFÉL_NÉV attribútum szerint, amely az ÜGYFÉL kulcsa, a BANKBETÉT-ben pedig idegen kulcs. Tegyük fel, hogy a két relációról a következő katalógusinformációk állnak rendelkezésre:

- $n_{\text{ÜGYFÉL}} = 10\ 000$
- $f_{\text{ÜGYFÉL}} = 25$ ($b_{\text{ÜGYFÉL}} = ?$)
- $n_{\text{BANKBETÉT}} = 5\ 000$
- $f_{\text{BANKBETÉT}} = 50$, ($b_{\text{BANKBETÉT}} = ?$)
- $V(\text{ÜGYFÉL\_NÉV}, \text{BANKBETÉT}) = 2\ 500$, ($SC(\text{ÜGYFÉL\_NÉV}, \text{BANKBETÉT}) = ?$)

Hány olyan ügyfél van, akinek nincsen bankbetétje?
Mekkora a BANKBETÉT és az ÜGYFÉL természetes illesztésének mérete, ha egyetlen közös attribútumuk az ÜGYFÉL_NÉV?
Általánosítsuk a feladatot az alábbi esetekre (R és S az illesztendő relációk sémái, természetes illesztéssel)!

- $R \cap S = \emptyset$.
- $R \cap S$ az R reláció kulcsa.
- $R \cap S \neq \emptyset$ egyik relációs sémának sem kulcsa.

3. feladat: hash join költsége

Számítsd ki a „hashjoin” algoritmussal végrehajtott join költségét, ha vödrös hashelést alkalmazunk! A hash függvény egyenletes eloszlással képezi le a kulcsokat az értékkészletére. Hogyan érdemes a join-t végrehajtani? A blokkméret nettó 2 000 bájt, a hashtábla szokás szerint elfér a memóriában.

- R reláció: 120 000 rekord, rekordhossz 150 bájt, kulcs 12 bájt, mutató 8 bájt, a hash tábla mérete 10 000 bájt.
- S reláció: 10 000 rekord, rekordhossz 250 bájt, kulcs 15 bájt, mutató 8 bájt a hash tábla mérete 1 000 bájt.

4. feladat: index-alapú illesztés

Számítsd ki az illesztés költségét, ha elsődleges, B*-fa struktúrájú indexeket használhatunk a join attribútumok szerinti rekordelérésre! A blokkméret nettó 4 000 bájt. Melyik reláció legyen a külső hurokban? Hányszoros válaszidőt kapunk, ha az optimalizáló rosszul dönt?

- R reláció: 140 000 rekord, rekordhossz 140 bájt, kulcs 10 bájt, mutató 4 bájt.
- S reláció: 15 000 rekord, rekordhossz 300 bájt, kulcs 6 bájt, mutató 4 bájt.

Gondolkodtató kérdések

1. Relációanalízis feladat: A relációs lekérdezések költség alapú optimalizásakor az ekvivalens alakokat előállító szabályoknak nincs jelentősége, mert csak a heurisztikus (vagy szabály alapú) optimalizálás alapszik átalakítási szabályokon.
2. Vizsgáld meg a blokkalapú egymásba ágyazott ciklikus illesztés és az egymásba ágyazott ciklikus illesztés algoritmusokban: a legbelső ciklusmag hányszor fut le worst-case esetben. Vesd össze az eredményt a két algoritmus worst-case költségével, és magyarázd meg a tapasztaltakat.
3. A természetes illesztésnél előbb „érdemes” végrehajtani az egy relációt érintő szelekció műveleteket, és ezt meg is tehetjük: ekvivalens algebrai alakot kapunk, ha lentebb (az alaprelációk irányába) süllyesztjük a szelekciókat. Megtehető-e ugyanez külső illesztés esetén? Mitől függ, hogy megtehető-e?
4. Lehet-e értelme olyan alapos költségbecslést és optimalizációt végezni, amelynek a költsége eléri/meghaladja a lekérdezés legköltségesebb szóba jövő végrehajtási módja költségét?

5. Melyek azok a műveletek, ahol elsődleges index használata olcsóbb végrehajtást eredményez, mint egy egyszerű index használata?
6. Mutass példát arra, amikor az elsődleges index használata ront a teljesítményen!
7. Mutass példát, amikor a lineáris keresés olcsóbb, mint az indexelt keresés? Próbáld általánosan megfogalmazni az eredményt!
8. Mi a biztos jele annak, hogy adatbázisunkból a szekvenciálisan generált ID nevű, elsődleges kulcsmezőt törölni kell? Mit vesztünk azzal, ha nem töröljük?
9. A relációs lekérdezések végrehajtási folyamatában szereplő optimalizálás bemenete miért relációs algebrajellegű kifejezés, ahelyett hogy valamilyen kalkulus-kifejezés lenne?
10. Az „index alapú illesztés" feladatban a relációs sémák kulcsait különböző hosszon tároltuk, azonban a természetes illesztés miatt rajtuk egyenlőségvizsgálatot kellett végezni. Lehetséges ilyen eset? Mutass példát rá, ha lehetséges, és indokold, ha nem!
11. Az „index alapú illesztés" feladatban erre vonatkozó információ hiányában feltételeztük, hogy az indexelt kulcsok egyedi értékűek a sémára illeszkedő relációban. Hogyan változik a join költsége/memóriaigénye, ha az egyes relációkban
    - $SC(Kulcs, R) < f_R$?
    - $SC(Kulcs, R)$ tetszőleges?

Gondolkodtató példa: több attribútumos indexek

A „D" Bank napi működését támogató adatbázisban a tranzakciókat a TRANZAKCIÓ(ÜGYFÉLSZÁM, DÁTUM, IDŐPONT, TÍPUS, PARTNER_SZÁMLASZÁMA, ÖSSZEG) sémára illeszkedő, heap szervezésű relációban tárolják. A tranzakció-rekordok nem módosíthatók. A napi működés középpontjában az ügyfél áll: a banki tranzakciók, mint pl. a vásárlás, átutalás, ügyfélmúlt lekérdezése (pl. netbankon max. 90 napra vonatkozóan) hozzá kapcsolódnak, ezért a működést az (ÜGYFÉLSZÁM, DÁTUM) attribútum párra épített elsődleges, B*-fa struktúrájú index támogatja.
Elemzéshez a bank indulásától fogva előállítják az adatbázis egy másolatát, melyben a tárolási struktúra megegyezik az on-line rendszerrel, de az új rekordok betöltése napi szinten történik: mindig az előző napi új rekordok kerülnek áttöltésre.
Az első időszakban flottul ment a dolog: a hajnali 3:03 órakor elindított áttöltés pillanatok alatt lefutott. Az első üzleti év végén azt tapasztalták, hogy 7 óra körül készül el az áttöltés. Fél év múlva csak délelőtt 11 óra után állt készen az elemzőrendszer. Az elemzési osztály vezetője egy tankönyvet lobogtatva ment a vezérigazgatóhoz, amiben azt olvasta (környezetéből kiragadott idézet): „a nagy mennyiségű adat batch jellegű betöltését tipikusan jelentősen lassítja, ha index van a cél táblán". Nosza, a vezérigazgató kiutalta a prémiumot az osztályvezetőnek, és utasította az informatikusokat az elemzőrendszerbeli index lekapcsolására.
A bank ügyfélköre kezdettől fogva egyenletesen emelkedett az első év végéig, amikor megállt a növekedés: pontosan 160 000 ügyfele van a banknak. Az ügyfél-lemorzsolódás elhanyagolható. Egy tipikus ügyfél naponta 4 tranzakciót végez. A blokkméret nettó 4 000 bájt, egy tranzakciós rekord 100 bájt, amiből az ügyfélszám 8 bájt, a dátum 4 bájt. A rendszer 64 bites mutatókat használ. Egy blokkművelet időigénye 5 ms.

- Milyen hatással van az index lekapcsolása az elemzők által futtatott lekérdezésekre, amelyek tipikusan 1 havi összesítő számításokat végeznek? (azaz mennyi idő alatt futott le az index lekapcsolása előtt egy ilyen lekérdezés, és hogyan változott az időigény az index lekapcsolása után)
- Számítsuk ki, vajon segít-e az elemzőrendszerbeli index lekapcsolása a bank áttöltési problémáján.
- Javasoljunk jobb megoldást a bank problémájára, és számítsuk ki, hogy változatlan ügyfélszám és adatbázis-kezelő rendszer mellett hány évig biztosítja a megoldás, hogy 7 óra körül készen álljon az áttöltés?
- Most, hogy a korszakalkotó javaslat után az elemzőrendszer mindig készen áll a reggeli munkakezdésre, optimalizáljuk azt is a korábban említett lekérdezési profilra úgy, hogy ne romoljon jelentősen az áttöltési teljesítmény!